Max-flow min-cut theorem

From Wikipedia, the free encyclopedia

In optimization theory, the **max-flow min-cut theorem** states that in a flow network, the maximum amount of flow passing from the *source* to the *sink* is equal to the total weight of the edges in the minimum cut, i.e. the smallest total weight of the edges which if removed would disconnect the source from the sink.

The **max-flow min-cut theorem** is a special case of the duality theorem for linear programs and can be used to derive Menger's theorem and the König–Egerváry theorem.

Contents

Definitions and statement

Let $N = (V, E)$ be a network (directed graph) with s and $t \in V$ being the source and the sink of N respectively.

Maximum flow

Definition. The **capacity** of an edge is a mapping $c : E \to \mathbf{R}^+$, denoted by c_{uv} or $c(u, v)$. It represents the maximum amount of flow that can pass through an edge.

Definition. A **flow** is a mapping $f : E \to \mathbf{R}^+$, denoted by f_{uv} or $f(u, v)$, subject to the following two constraints:

1. Capacity Constraint:
   $\forall (u,v) \in E: \qquad f_{uv} \le c_{uv}$
2. Conservation of Flows:
   $\forall v \in V \setminus \{s,t\}: \qquad \sum_{\{u:(u,v)\in E\}} f_{uv} = \sum_{\{u:(v,u)\in E\}} f_{vu}.$

Definition. The **value of flow** is defined by

$$|f| = \sum_{v\in V} f_{sv},$$

where s is the source of N. It represents the amount of flow passing from the source to the sink.

Maximum Flow Problem. Maximize $|f|$, that is, to route as much flow as possible from s to t.

Minimum cut

Definition. An **s-t cut** $C = (S, T)$ is a partition of V such that $s \in S$ and $t \in T$. The **cut-set** of C is the set

$$\{(u,v) \in E \ : \ u \in S, v \in T\}.$$

Note that if the edges in the cut-set of C are removed, $|f| = 0$.

Definition. The **capacity** of an *s-t cut* is defined by

$$c(S,T) = \sum_{(u,v)\in(S\times T)\cap E} c_{uv} = \sum_{(i,j)\in E} c_{ij} d_{ij},$$

where $d_{ij} = 1$ if $i \in S$ and $j \in T$, 0 otherwise.

Minimum s-t Cut Problem. Minimize $c(S, T)$, that is, to determine S and T such that the capacity of the *S-T cut* is minimal.

Statement

Max-flow min-cut theorem. The maximum value of an s-t flow is equal to the minimum capacity over all s-t cuts.

Linear program formulation

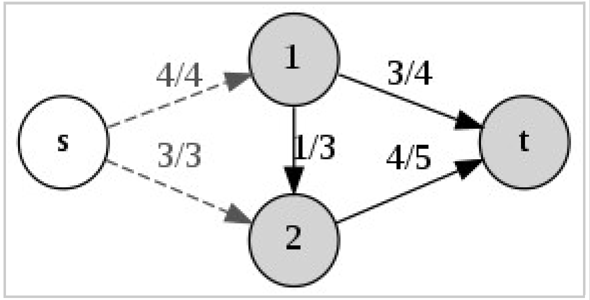
The max-flow problem and min-cut problem can be formulated as two primal-dual linear programs.

Max-flow (Primal)	Min-cut (Dual)
maximize $\lvert f \rvert = \nabla_s$ subject to $$\begin{array}{rcll} f_{ij} & \leq & c_{ij} & (i,j) \in E \\ \sum_{j:(j,i)\in E} f_{ji} - \sum_{j:(i,j)\in E} f_{ij} & \leq & 0 & i \in V, i \neq s,t \\ \nabla_s + \sum_{j:(j,s)\in E} f_{js} - \sum_{j:(s,j)\in E} f_{sj} & \leq & 0 & \\ -\nabla_s + \sum_{j:(j,t)\in E} f_{jt} - \sum_{j:(t,j)\in E} f_{tj} & \leq & 0 & \\ f_{ij} & \geq & 0 & (i,j) \in E \end{array}$$	minimize $\sum_{(i,j)\in E} c_{ij} d_{ij}$ subject to $$\begin{array}{rcll} d_{ij} - p_i + p_j & \geq & 0 & (i,j) \in E \\ p_s - p_t & \geq & 1 & \\ p_i & \geq & 0 & i \in V \\ d_{ij} & \geq & 0 & (i,j) \in E \end{array}$$

Note that for the given s-t cut $C = (S, T)$ if $i \in S$ then $p_i = 1$ and 0 otherwise. Therefore, p_s should be 1 and p_t should be zero. The equality in the **max-flow min-cut theorem** follows from the strong duality theorem in linear programming, which states that if the primal program has an optimal solution, *x**, then the dual program also has an optimal solution, *y**, such that the optimal values formed by the two solutions are equal.

Example

The figure on the right is a network having a value of flow of 7. The vertex in white and the vertices in grey form the subsets S and T of an s-t cut, whose cut-set contains the dashed edges. Since the capacity of the s-t cut is 7, which equals to the value of flow, the max-flow min-cut theorem tells us that the value of flow and the capacity of the s-t cut are both optimal in this network.



A network with the value of flow equal to the capacity of an s-t cut

Application

Generalized max-flow min-cut theorem

In addition to edge capacity, consider there is capacity at each vertex, that is, a mapping $c : V \to \mathbf{R}^+$, denoted by $c(v)$, such that the flow f has to satisfy not only the capacity constraint and the conservation of flows, but also the vertex capacity constraint

$$\forall v \in V \setminus \{s,t\} : \qquad \sum\nolimits_{\{i \in V | (iv) \in E\}} f_{iv} \le c(v).$$

In other words, the amount of *flow* passing through a vertex cannot exceed its capacity. Define an *s-t cut* to be the set of vertices and edges such that for any path from *s* to *t*, the path contains a member of the cut. In this case, the *capacity of the cut* is the sum the capacity of each edge and vertex in it.

In this new definition, the **generalized max-flow min-cut theorem** states that the maximum value of an s-t flow is equal to the minimum capacity of an s-t cut in the new sense.

Menger's theorem

In the undirected edge-disjoint paths problem, we are given an undirected graph $G = (V, E)$ and two vertices s and t, and we have to find the maximum number of edge-disjoint s-t paths in G.

The **Menger's theorem** states that the maximum number of edge-disjoint s-t paths in an undirected graph is equal to the minimum number of edges in an s-t cut-set.

Project selection problem

In the project selection problem, there are n projects and m machines. Each project p_i yields revenue $r(p_i)$ and each machine q_j costs $c(q_j)$ to purchase. Each project requires a number of machines and each machine can be shared by several projects. The problem is to determine which projects and machines should be selected and purchased respectively, so that the profit is maximized.

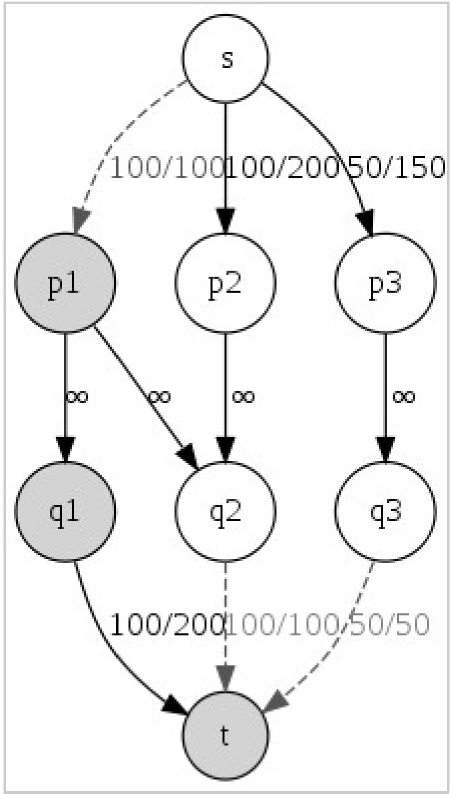
Let P be the set of projects *not* selected and Q be the set of machines purchased, then the problem can be formulated as,

$$\max\{g\} = \sum_i r(p_i) - \sum_{p_i \in P} r(p_i) - \sum_{q_j \in Q} c(q_j).$$

Since the first term does not depend on the choice of P and Q, this maximization problem can be formulated as a minimization problem instead, that is,

$$\min\{g'\} = \sum_{p_i \in P} r(p_i) + \sum_{q_j \in Q} c(q_j).$$

The above minimization problem can then be formulated as a minimum-cut problem by constructing a network, where the source is connected to the projects with capacity $r(p_i)$, and the sink is connected by the machines with capacity $c(q_j)$. An edge (p_i, q_j) with *infinite* capacity is added if project p_i requires machine q_j. The s-t cut-set represents the projects and machines in P and Q respectively. By the max-flow min-cut theorem, one can solve the problem as a maximum flow problem.



A network formulation of the project selection problem with the optimal solution

The figure on the right gives a network formulation of the following project selection problem:

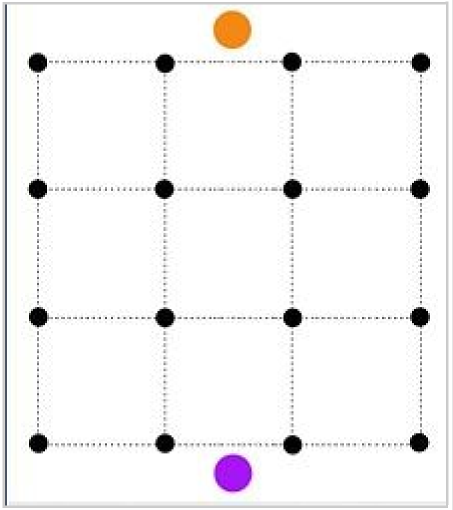
	Project $r(p_i)$	**Machine $c(q_j)$**	
1	100	200	Project 1 requires machines 1 and 2.
2	200	100	Project 2 requires machine 2.
3	150	50	Project 3 requires machine 3.

The minimum capacity of a s-t cut is 250 and the sum of the revenue of each project is 450; therefore the maximum profit g is $450 - 250 = 200$, by selecting projects p_2 and p_3.

The idea here is to 'flow' the project profits through the 'pipes' of the machine. If we cannot fill the pipe, the machine's return is less than its cost, and the min cut algorithm will find it cheaper to cut the project's profit edge instead of the machine's cost edge.

Image segmentation problem

In the image segmentation problem, there are n pixels. Each pixel i can be assigned a foreground value f_i or a background value b_i. There is a penalty of p_{ij} if pixels i, j are adjacent and have different assignments. The problem is to assign pixels to foreground or background such that the sum of their values minus the penalties is maximum.


Each black node denotes a pixel.

Let P be the set of pixels assigned to foreground and Q be the set of points assigned to background, then the problem can be formulated as,

$$\max\{g\} = \sum_{i\in P} f_i + \sum_{i\in Q} b_i - \sum_{i\in P, j\in Q \vee j\in P, i\in Q} p_{ij}.$$

This maximization problem can be formulated as a minimization problem instead, that is,

$$\min\{g'\} = \sum_{i\in P, j\in Q \vee j\in P, i\in Q} p_{ij}.$$

The above minimization problem can be formulated as a minimum-cut problem by constructing a network where the source (orange node) is connected to all the pixels with capacity f_i, and the sink (purple node) is connected by all the pixels with capacity b_i. Two edges (i, j) and (j, i) with p_{ij} capacity are added between two adjacent pixels. The s-t cut-set then represents the pixels assigned to the foreground in P and pixels assigned to background in Q.

History

The **max-flow min-cut theorem** was proven by P. Elias, A. Feinstein, and C.E. Shannon in 1956[1], and independently also by L.R. Ford, Jr. and D.R. Fulkerson in the same year[2].

Proof

Let $G = (V, E)$ be a network (directed graph) with s and t being the source and the sink of G respectively.

Consider the flow f computed for G by Ford–Fulkerson algorithm. In the residual graph (G_f) obtained for G (after the final flow assignment by Ford–Fulkerson algorithm), define two subsets of vertices as follows:

1. A: the set of vertices reachable from s in G_f
2. A^c: the set of remaining vertices i.e. $V - A$

Claim. $\text{value}(f) = c(A, A^c)$, where the **capacity** of an *s-t cut* is defined by

$$c(S,T) = \sum\nolimits_{(u,v)\in S\times T} c_{uv}\,.$$

Now, we know, $value(f) = f_{out}(A) - f_{in}(A)$ for any subset of vertices, A. Therefore, for $\text{value}(f) = c(A, A^c)$ we need:

- All *outgoing edges* from the cut must be **fully saturated**.
- All *incoming edges* to the cut must have **zero flow**.

To prove the above claim we consider two cases:

- In G, there exists an *outgoing edge* $(x,y), x \in A, y \in A^c$ such that it is not saturated, i.e., $f(x, y) < c_{xy}$. This implies, that there exists a **forward edge** from x to y in G_f, therefore there exists a path from s to y in G_f, which is a contradiction. Hence, any outgoing edge (x, y) is fully saturated.
- In G, there exists an *incoming edge* $(y,x), x \in A, y \in A^c$ such that it carries some non-zero flow, i.e., $f(x, y) > 0$. This implies, that there exists a **backward edge** from x to y in G_f, therefore there exists a path from s to y in G_f, which is again a contradiction. Hence, any incoming edge (x, y) must have zero flow.

Both of the above statements prove that the capacity of cut obtained in the above described manner is equal to the flow obtained in the network. Also, the flow was obtained by Ford-Fulkerson algorithm, so it is the max-flow of the network as well.

> Also, since *any flow in the network is always less than or equal to capacity of every cut possible in a network*, the above described cut is also the min-cut which obtains the max-flow.

See also

- Linear programming

- Maximum flow
- Minimum cut
- Flow network
- Edmonds–Karp algorithm
- Ford–Fulkerson algorithm
- Approximate max-flow min-cut theorem

References

1. Eugene Lawler (2001). "4.5. Combinatorial Implications of Max-Flow Min-Cut Theorem, 4.6. Linear Programming Interpretation of Max-Flow Min-Cut Theorem". *Combinatorial Optimization: Networks and Matroids*. Dover. pp. 117–120. ISBN 0-486-41453-1.
2. Christos H. Papadimitriou, Kenneth Steiglitz (1998). "6.1 The Max-Flow, Min-Cut Theorem". *Combinatorial Optimization: Algorithms and Complexity*. Dover. pp. 120–128. ISBN 0-486-40258-4.
3. Vijay V. Vazirani (2004). "12. Introduction to LP-Duality". *Approximation Algorithms*. Springer. pp. 93–100. ISBN 3-540-65367-8.

Categories: Combinatorial optimization | Theorems in graph theory | Network flow